

第5章 Redis主从集群

为了避免Redis的单点故障问题，我们可以搭建一个Redis集群，将数据备份到集群中的其它节点上。若一个Redis节点宕机，则由集群中的其它节点顶上。

1 主从集群搭建与启动

Redis的主从集群是一个“一主多从”的读写分离集群。集群中的Master节点负责处理客户端的读写请求，而Slave节点仅能处理客户端的读请求。之所以要将集群搭建为读写分离模式，主要原因是，对于数据库集群，写操作压力一般都较小，压力大多数来自于读操作请求。所以，只有一个节点负责处理写操作请求即可。

集群搭建

在采用单线程IO模型时，为了提高处理器的利用率，一般会在一个主机中安装多台Redis，构建一个Redis主从伪集群。当然，搭建伪集群的另一个场景是，在学习Redis，而学习用主机内存不足以创建多个虚拟机。

下面要搭建的读写分离伪集群包含一个Master与两个Slave。它们的端口号分别是：6380、6381、6382。

复制redis.conf

在redis安装目录中mkdir一个目录，名称随意。这里命名为cluster。然后将redis.conf文件复制到cluster目录中。该文件后面会被其它配置文件包含，所以该文件中需要设置每个Redis节点相同的公共的属性。

新建redis6380.conf

新建一个redis配置文件redis6380.conf，该配置文件中的Redis端口号为6380。

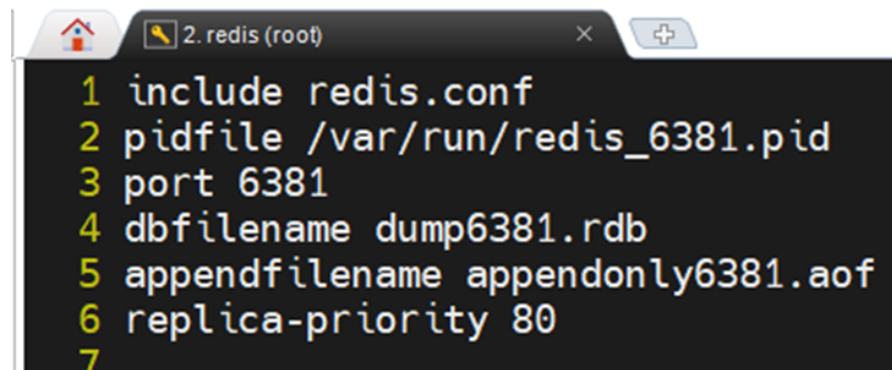


```
1 include redis.conf
2 pidfile /var/run/redis_6380.pid
3 port 6380
4 dbfilename dump6380.rdb
5 appendfilename appendonly6380.aof
6 replica-priority 90
7
```

再复制出两个conf文件

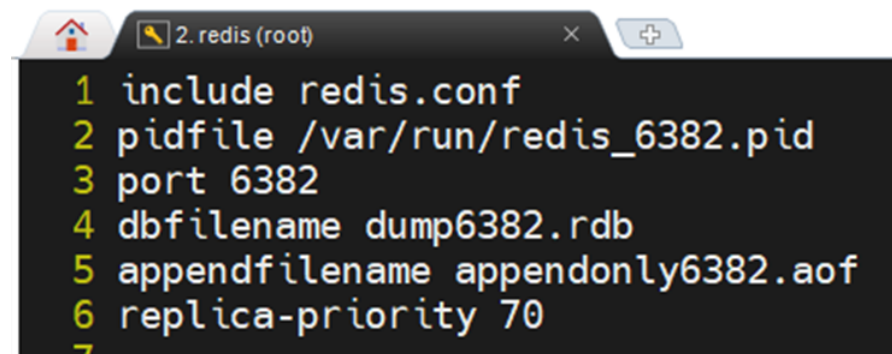
再使用redis6380.conf复制出两个conf文件：redis6381.conf与redis6382.conf。然后修改其中的内容。

修改redis6381.conf的内容如下：

A terminal window titled '2. redis (root)' with a search icon and a plus icon. It displays the configuration for redis6381.conf with line numbers 1 through 7.

```
1 include redis.conf
2 pidfile /var/run/redis_6381.pid
3 port 6381
4 dbfilename dump6381.rdb
5 appendfilename appendonly6381.aof
6 replica-priority 80
7
```

修改redis6382.conf的内容如下：

A terminal window titled '2. redis (root)' with a search icon and a plus icon. It displays the configuration for redis6382.conf with line numbers 1 through 7.

```
1 include redis.conf
2 pidfile /var/run/redis_6382.pid
3 port 6382
4 dbfilename dump6382.rdb
5 appendfilename appendonly6382.aof
6 replica-priority 70
7
```

集群的设置与启动

启动三台Redis

分别使用redis6380.conf、redis6381.conf与redis6382.conf三个配置文件启动三台Redis。

再打开三个会话框，分别使用客户端连接三台Redis。然后通过slaveof命令，指定6380的Redis为Master。

```
[root@redisOS cluster]#  
[root@redisOS ~]# redis-cli -p 6381  
127.0.0.1:6381> slaveof 127.0.0.1 6380  
OK  
127.0.0.1:6381>  
[root@redisOS ~]# redis-cli -p 6380  
127.0.0.1:6380> slaveof 127.0.0.1 6382  
OK  
127.0.0.1:6380>  
[root@redisOS ~]# redis-cli -p 6382  
127.0.0.1:6382> slaveof 127.0.0.1 6380  
OK  
127.0.0.1:6382>
```

查看状态信息

通过info replication命令可查看当前连接的Redis的状态信息。

```
[root@redisOS cluster]#  
127.0.0.1:6380> info replication  
# Replication  
role:master  
connected_slaves:2  
slave0:ip=127.0.0.1,port=6381,state=online,offset=602,lag=0  
slave1:ip=127.0.0.1,port=6382,state=online,offset=602,lag=0  
master_failover_state:no-failover  
master_replid:3f9ef35fe7489ab1738cdaa15cd91c4965bfa20  
127.0.0.1:6381> info replication  
# Replication  
role:slave  
master_host:127.0.0.1  
master_port:6380  
master_link_status:up  
master_last_io_seconds_ago:8  
master_sync_in_progress:0  
slave_read_repl_offset:630  
slave_repl_offset:630  
slave_priority:100  
127.0.0.1:6382> info replication  
# Replication  
role:slave  
master_host:127.0.0.1  
master_port:6380  
master_link_status:up  
master_last_io_seconds_ago:4  
master_sync_in_progress:0  
slave_read_repl_offset:658  
slave_repl_offset:658  
slave_priority:100
```

分级管理

若Redis主从集群中的Slave较多时，它们的数据同步过程会对Master形成较大的性能压力。此时可以对这些Slave进行分级管理。

设置方式很简单，只需要让低级别Slave指定其slaveof的主机为其上一级Slave即可。不过，上一级Slave的状态仍为Slave，只不过，其是更上一级的Slave。

例如，指定6382主机为6381主机的Slave，而6381主机仍为真正的Master的Slave。

```
[root@redis05 cluster]#  
  
127.0.0.1:6380> info replication  
# Replication  
role:master  
connected_slaves:1  
slave0:ip=127.0.0.1,port=6381,state=online,offset=12973,lag=0  
master_failover_state:no-failover  
master_replid:3f9ef35fe7489ab1738cdaaa15cd91c4965bfa20  
master_replid2:00000000000000000000000000000000  
  
127.0.0.1:6381> info replication  
# Replication  
role:slave  
master_host:127.0.0.1  
master_port:6380  
master_link_status:up  
master_last_io_seconds_ago:2  
master_sync_in_progress:0  
slave_read_repl_offset:12959  
slave_repl_offset:12959  
slave_priority:100  
  
127.0.0.1:6382> slaveof 127.0.0.1 6381  
OK  
127.0.0.1:6382> info replication  
# Replication  
role:slave  
master_host:127.0.0.1  
master_port:6381  
master_link_status:up  
master_last_io_seconds_ago:6  
master_sync_in_progress:0  
slave_read_repl_offset:12917
```

此时会发现，Master的Slave只有6381一个主机。

容灾冷处理

在Master/Slave的Redis集群中，若Master出现宕机怎么办呢？有两种处理方式，一种是通过手工角色调整，使Slave晋升为Master的冷处理；一种是使用哨兵模式，实现Redis集群的高可用HA，即热处理。

无论Master是否宕机，Slave都可通过slaveof no one将自己由Slave晋升为Master。如果其原本就有下一级的Slave，那么，其就直接变为了这些Slave的真正的Master了。而原来的Master也会失去这个原来的Slave。

```
[root@redis05 cluster]#  
  
127.0.0.1:6380> info replication  
# Replication  
role:master  
connected_slaves:0  
master_failover_state:no-failover  
master_replid:3f9ef35fe7489ab1738cdaaa15cd91c4965bfa20  
master_replid2:00000000000000000000000000000000  
master_repl_offset:13449  
second_repl_offset:-1  
  
127.0.0.1:6381> slaveof no one  
OK  
127.0.0.1:6381> info replication  
# Replication  
role:master  
connected_slaves:1  
slave0:ip=127.0.0.1,port=6382,state=online,offset=13309,lag=0  
master_failover_state:no-failover  
master_replid:0da63c95c17f87aecab8e25ec6cb81264aeae051
```

2 哨兵机制实现

简介

对于Master宕机后的冷处理方式是无法实现高可用的。Redis从2.6版本开始提供了高可用的解决方案——Sentinel哨兵机制。在集群中再引入一个节点，该节点充当Sentinel哨兵，用于监视Master的运行状态，并在Master宕机后自动指定一个Slave作为新的Master。整个过程无需人工参与，完全由哨兵自动完成。

不过，此时的Sentinel哨兵又成为了一个单点故障点：若哨兵发生宕机，整个集群将瘫痪。所以为了解决Sentinel的单点问题，又要为Sentinel创建一个集群，即Sentinel哨兵集群。一个哨兵的宕机，将不会影响到Redis集群的运行。

那么这些Sentinel哨兵是如何工作的呢？Sentinel是如何知道其监视的Master状态的呢？每个Sentinel都会定时向Master发送心跳，如果Master在有效时间内向它们都进行了响应，则说明Master是“活着的”。如果Sentinel中有quorum个哨兵没有收到响应，那么就认为Master已经宕机，然后会有一个Sentinel做Failover故障转移。即将原来的某一个Slave晋升为Master。

Redis高可用集群搭建

在“不差钱”的情况下，可以让Sentinel占用独立的主机，即在Redis主机上只启动Redis进程，在Sentinel主机上只启动Sentinel进程。下面要搭建一个“一主二从三哨兵”的高可用伪集群，即这些角色全部安装运行在一台主机上。“一主二从”使用前面的主从集群，下面仅搭建一个Sentinel伪集群。

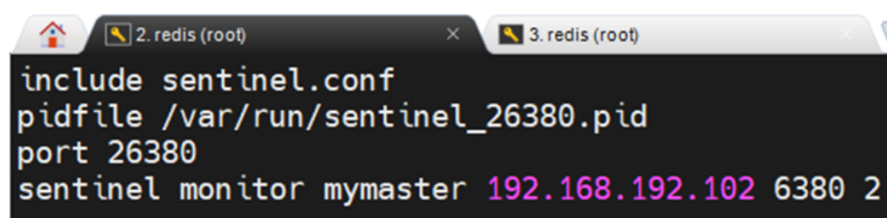
角色	端口号	角色	端口号
master	6380	sentinel	26380
slave	6381	sentinel	26381
slave	6382	sentinel	26382

复制sentinel.conf

将Redis安装目录中的sentinel.conf文件复制到cluster目录中。该配置文件中用于存放一些sentinel集群中的一些公共配置。

新建sentinel26380.conf

在Redis安装目录下的cluster目录中新建sentinel26380.conf文件作为Sentinel的配置文件，并在其中键入如下内容：



```
include sentinel.conf
pidfile /var/run/sentinel_26380.pid
port 26380
sentinel monitor mymaster 192.168.192.102 6380 2
```

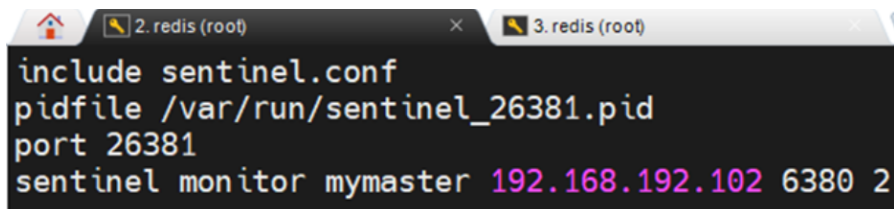
sentinel monitor属性用于指定当前监控的master的IP与Port，同时为集群中master指定一个名称mymaster，以方便其它属性使用。

最后的2是参数quorum的值，quorum有两个用途。一个是只有当quorum个sentinel都认为当前master宕机了才能开启故障转移。另一个用途与sentinel的Leader选举有关。要求中至少要有 $\max(\text{quorum}, \text{sentinelNum}/2+1)$ 个sentinel参与，选举才能进行。

再复制两个conf文件

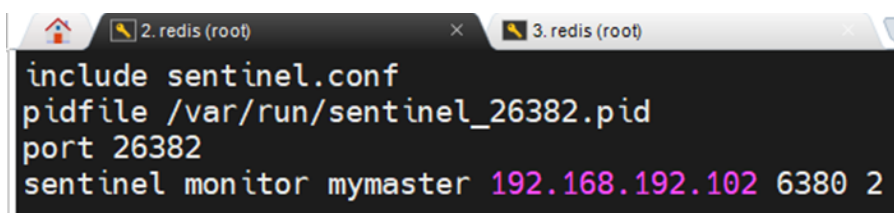
再使用sentinel26380.conf复制出两个conf文件：sentinel26381.conf与sentinel26382.conf。然后修改其中的内容。

修改sentinel26381.conf。



```
include sentinel.conf
pidfile /var/run/sentinel_26381.pid
port 26381
sentinel monitor mymaster 192.168.192.102 6380 2
```

修改sentinel26382.conf。

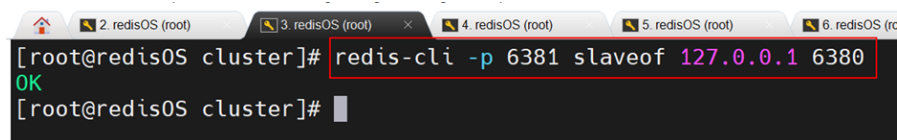


```
include sentinel.conf
pidfile /var/run/sentinel_26382.pid
port 26382
sentinel monitor mymaster 192.168.192.102 6380 2
```

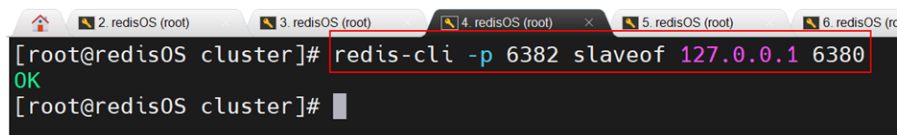
Redis高可用集群启动

启动并关联Redis集群

首先要启动三台Redis，然后再通过slaveof关联它们。



```
[root@redisOS cluster]# redis-cli -p 6381 slaveof 127.0.0.1 6380
OK
[root@redisOS cluster]#
```



```
[root@redisOS cluster]# redis-cli -p 6382 slaveof 127.0.0.1 6380
OK
[root@redisOS cluster]#
```

启动Sentinel集群

由于redis-server与redis-sentinel命令本质上是同一个命令，所以使用这两个命令均可启动Sentinel。

- 方式一，使用redis-sentinel命令：redis-sentinel sentinel26380.conf
- 方式二，使用redis-server命令：redis-server sentinel26380.conf --sentinel


```
[root@redis cluster]# redis-sentinel sentinel26380.conf
7671:X 11 Sep 2022 18:03:42.961 # 000000000000 Redis is starting 000000000000
7671:X 11 Sep 2022 18:03:42.961 # Redis version=7.0.4, bits=64, commit=00000000, modified=0, pid=7671, just s
7671:X 11 Sep 2022 18:03:42.961 # Configuration loaded
7671:X 11 Sep 2022 18:03:42.962 * Increased maximum number of open files to 10032 (it was originally set to 1024)
7671:X 11 Sep 2022 18:03:42.962 * monotonic clock: POSIX clock_gettime

Redis 7.0.4 (00000000/0) 64 bit

Running in sentinel mode
Port: 26380
PID: 7671

https://redis.io

7671:X 11 Sep 2022 18:03:42.965 * Sentinel new configuration saved on disk
7671:X 11 Sep 2022 18:03:42.965 # Sentinel ID is f90bfe3a48fe54ae0f507198ce65a168ff57d842
7671:X 11 Sep 2022 18:03:42.965 # +monitor master mymaster 192.168.192.102 6380 quorum 2
7671:X 11 Sep 2022 18:03:42.966 * +slave slave 127.0.0.1:6381 127.0.0.1 6381 @ mymaster 192.168.192.102 6380
7671:X 11 Sep 2022 18:03:42.967 * Sentinel new configuration saved on disk
7671:X 11 Sep 2022 18:03:42.967 * +slave slave 127.0.0.1:6382 127.0.0.1 6382 @ mymaster 192.168.192.102 6380
7671:X 11 Sep 2022 18:03:42.968 * Sentinel new configuration saved on disk
```

```
[root@redis cluster]# redis-sentinel sentinel26381.conf
```

```
[root@redis cluster]# redis-sentinel sentinel26382.conf
```

查看Sentinel信息

运行中的Sentinel就是一个特殊Redis，其也可以通过客户端连接，然后通过info sentinel来查看当前连接的Sentinel的信息。

```
[root@redis cluster]# redis-cli -p 26380 info sentinel
# Sentinel
sentinel_masters:1
sentinel_tilt:0
sentinel_tilt_since_seconds:-1
sentinel_running_scripts:0
sentinel_scripts_queue_length:0
sentinel_simulate_failure_flags:0
master0:name=mymaster,status=ok,address=192.168.192.102:6380,slaves=2,sentinels=3
[root@redis cluster]#
[root@redis cluster]# redis-cli -p 26381 info sentinel
# Sentinel
sentinel_masters:1
sentinel_tilt:0
sentinel_tilt_since_seconds:-1
sentinel_running_scripts:0
sentinel_scripts_queue_length:0
sentinel_simulate_failure_flags:0
master0:name=mymaster,status=ok,address=192.168.192.102:6380,slaves=2,sentinels=3
[root@redis cluster]#
[root@redis cluster]# redis-cli -p 26382 info sentinel
# Sentinel
sentinel_masters:1
sentinel_tilt:0
sentinel_tilt_since_seconds:-1
sentinel_running_scripts:0
sentinel_scripts_queue_length:0
sentinel_simulate_failure_flags:0
master0:name=mymaster,status=ok,address=192.168.192.102:6380,slaves=2,sentinels=3
[root@redis cluster]#
```

